N92-14211

A Bandwidth Efficient Coding Scheme for the Hubble Space Telescope*

Steven S. Pietrobon
University of South Australia
The Levels, South Australia 5095
Australia

Daniel J. Costello, Jr.
University of Notre Dame
Notre Dame, Indiana 46556
U.S.A.

1 SUMMARY

As a demonstration of the performance capabilities of trellis codes using multidimensional signal sets, a Viterbi decoder for one of the codes in [1] was designed. The choice of code was based on two factors.

The first factor was its application as a possible replacement for the coding scheme currently used on the Hubble Space Telescope (HST). The HST at present uses the rate 1/3 $\nu = 6$ (with $2^{\nu} = 64$ states) convolutional code with BPSK modulation. With the modulator restricted to 3 Msym/s, this implies a data rate of only 1 Mbit/s, since the bandwidth efficiency $K = 1/3$ bit/sym. This is a very bandwidth inefficient scheme, although the system has the advantage of simplicity and large coding gain.

The basic requirement from NASA was for a scheme that has as large a K as possible. Since a satellite channel was being used, 8PSK modulation was selected. This allows a K of between 2 and 3 bit/sym. The next influencing factor was INTELSAT's intention of transmitting the SONET 155.52 Mbit/s standard data rate over the 72 MHz transponders on its satellites. This requires a bandwidth efficiency of around 2.5 bit/sym. A Reed-Solomon block code is used as an outer code to give very low bit error rates (BER).

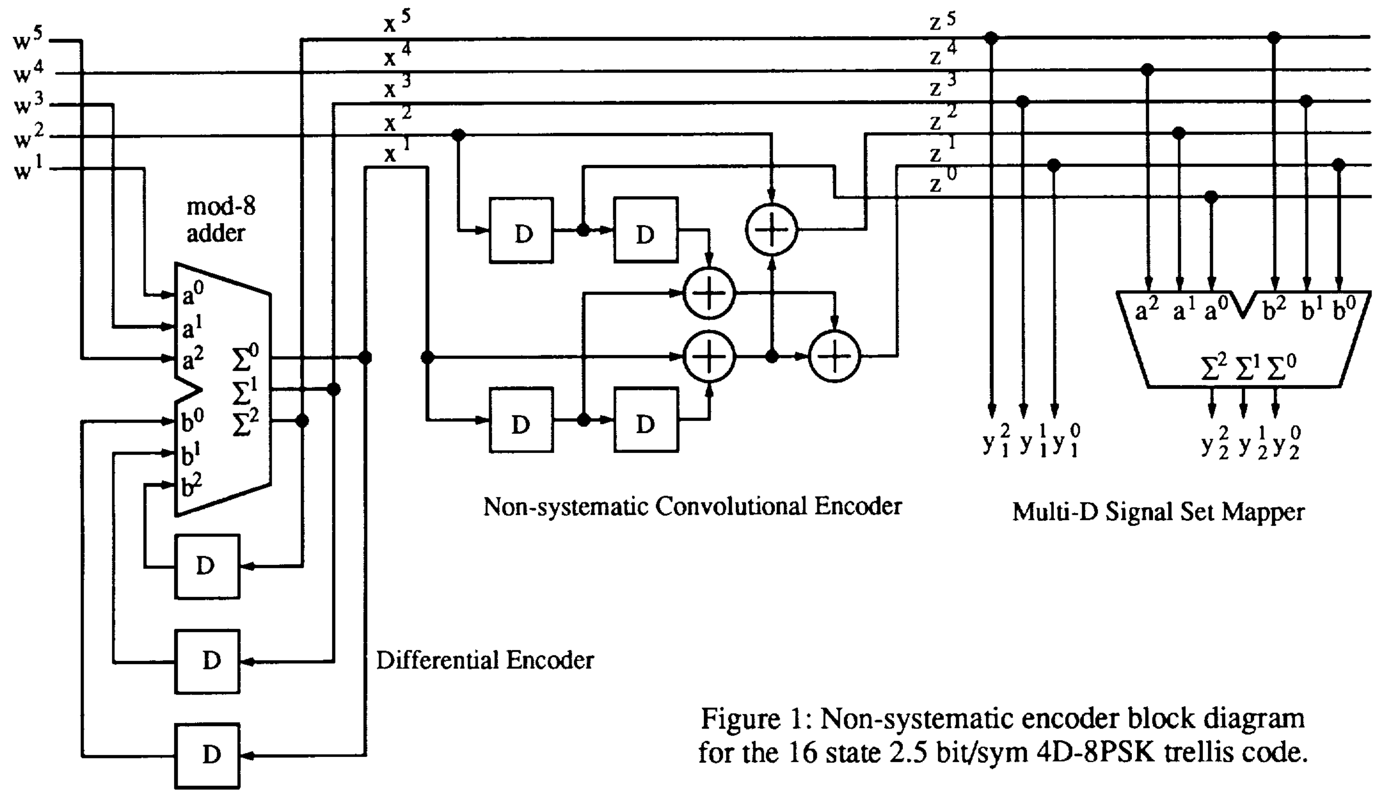
The 16 state rate 5/6, 2.5 bit/sym, 4D-8PSK trellis code from [1] was selected. This code has reasonable complexity and has a coding gain of 4.8 dB compared to uncoded 8PSK [2]. This trellis code also has the advantage that it is 45° rotationally invariant. This means that the decoder needs only to synchronise to one of the two naturally mapped 8PSK signals in the signal set.

2 ENCODER IMPLEMENTATION

At first, a systematic encoder was used in the design. However, it was found that in designing a Viterbi decoder, it would be simpler if a non-systematic convolutional encoder was used. This is because the state transitions in a non-systematic encoder are highly structured, compared with the almost "random" transitions of a systematic encoder.

To convert the systematic encoder to a non-systematic form, the technique described in [3] is used. This method uses the fact that the impulse response of each shift register in a non-systematic encoder will produce output sequences that are equivalent to the generator polynomials. Since a systematic encoder must also produce the same sequences, it is relatively easy to find $\tilde{k}$ linearly independent output sequences from a systematic encoder that

*This work was supported in part by NASA Grant NAG5-557 and in part by OTC Limited under Project 1662.



w^5
w^4
w^3
w^2
w^1
mod-8 adder
a^0 a^1 a^2 b^0 b^1 b^2
Σ^0 Σ^1 Σ^2
x^5 x^4 x^3 x^2 x^1
z^5 z^4 z^3 z^2 z^1 z^0
D
a^2 a^1 a^0 b^2 b^1 b^0
Σ^2 Σ^1 Σ^0
y_1^2 y_1^1 y_1^0
y_2^2 y_2^1 y_2^0
Non-systematic Convolutional Encoder
Multi-D Signal Set Mapper
Differential Encoder

Figure 1: Non-systematic encoder block diagram for the 16 state 2.5 bit/sym 4D-8PSK trellis code.

can be used as generators of a non-systematic encoder.

There is usually more than one set of possible generator polynomials. The polynomials are chosen so that the inputs $x^1(D)$ and $x^2(D)$ are affected by a 45° phase rotation in the same way as in a systematic encoder. Thus, the differential encoder for the systematic code can also be used for the non-systematic encoder. The non-systematic encoder equations that were found for the 4D-8PSK code are

$$z^2(D) = x^2(D) \oplus (D^2 \oplus 1)x^1(D), \quad (1a)$$

$$z^1(D) = D^2x^2(D) \oplus (D^2 \oplus D \oplus 1)x^1(D), \quad (1b)$$

$$z^0(D) = Dx^2(D). \quad (1c)$$

Figure 1 illustrates the new non-systematic encoder. After a 45° phase rotation, we have $z_r^2(D) = z^2(D)$, $z_r^1(D) = z^1(D) \oplus 1(D)$, and $z_r^0(D) = z^0(D)$. Rotating the equations in (1) gives $x_r^2(D) = x^2(D)$ and $x_r^1(D) = x^1(D) \oplus 1(D)$, the same as for the systematic encoder.

The encoder uses a Phase Locked Loop (PLL) to generate the two times clock for transmitting the two 2D symbols. This PLL is based on the 74HC4046 Integrated Circuit (IC). The encoder is able to accept data either serially or in five bit bytes.

3 DECODER IMPLEMENTATION

Due to the complexity of the decoder design, only a brief description is given here. As such, only the important design decisions are described.

To reduce the cost of the codec, a serial implementation of the decoder was chosen. That is, one clock cycle would be required for each state of the code. Since there are 16 states, at least 16 clock cycles are required to process each received 4D point. As will be described in more detail later, an extra seven clock cycles are required for start-up purposes. Thus, a total of 23 clock cycles are required for each iteration of the Viterbi algorithm.

The technology and clock speed in our design is the same as used in another Viterbi decoder designed by the author [4]. This gave us greater confidence that

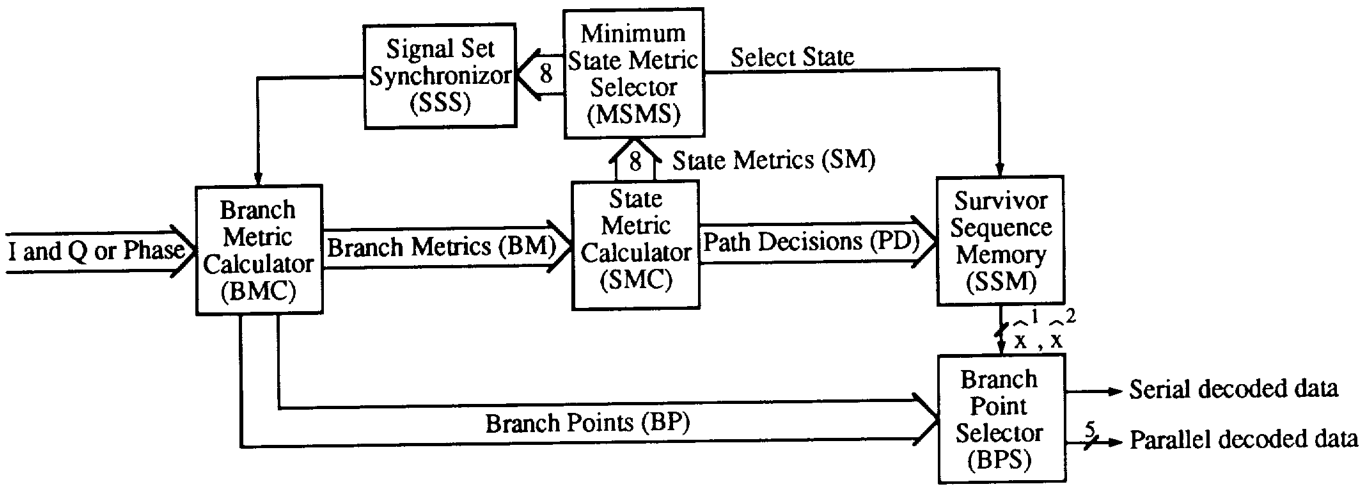

Signal Set Synchronizor (SSS)
8
Minimum State Metric Selector (MSMS)
Select State
8
State Metrics (SM)
I and Q or Phase
Branch Metric Calculator (BMC)
Branch Metrics (BM)
State Metric Calculator (SMC)
Path Decisions (PD)
Survivor Sequence Memory (SSM)
$\hat{x}^1, \hat{x}^2$
Branch Points (BP)
Branch Point Selector (BPS)
Serial decoded data
5
Parallel decoded data

Figure 2: Block diagram of a Viterbi decoder for the 16 state 2.5 bit/sym 4D-8PSK trellis code.

the design would work, even though the actual design is twice as complicated. Our design uses a 10 MHz clock (giving 100 ns clock cycles) and Schottky TTL logic for its ease of use and large variety of functions. The actual technologies used are 74LS (Low-power Schottky TTL) for non-time critical sections of the circuit and 74F (Advanced Schottky TTL) for time critical sections. Other technologies are used for functions not available in 74F or 74LS.

The decoder is operated asynchronously to the received data clock. This requires one of the seven extra clock cycles described above. Internally, the decoder operates synchronously to the 10 MHz clock. The decoder starts operation after detecting the first rising edge of the received 4D symbol clock. After 23 clock cycles, the decoder stops and waits for the next rising edge of the 4D symbol clock. This allows the decoder to operate at any data rate from 0 to 2.1 Mbit/s.

Each iteration of the Viterbi algorithm decodes five bits for each received 4D signal point (since the code rate is 5/6). The maximum 4D symbol rate of the decoder is the internal clock speed divided the number of clock cycles required to decode the five bits, i.e., 4.35×10^5 4D symbols per second. Therefore, the maximum bit rate of the decoder is 2.17 Mbit/s. For the HST, this code could achieve a data rate up to 7.5 Mbit/s. For actual use on the HST, it is intended that the decoder would be implemented on a VLSI chip, where the required decoding speed would be achieved.

There are six main sections in the Viterbi decoder. These are

- Branch Metric Calculator (BMC)
- State Metric Calculator (SMC)
- Survivor Sequence Memory (SSM)
- Signal Set Synchronisor (SSS)
- Minimum State Metric Selector (MSMS)
- Branch Point Selector (BPS)

Figure 2 illustrates a block diagram of the decoder. The above sections are described as follows.

3.1 Branch Metric Calculator

For each transition of the trellis there are 8 parallel paths (due to the three unchecked bits in the encoder). The BMC must determine which of the paths is closest to the received 4D signal point (the Branch Point (BP)) as well as the Branch Metric (BM) for this path. The BM can be calculated in a number of ways. The optimum BM's for AWGN channels with quantisation are log-likelihood metrics [4]. Alternatively, one could make an approximation based on the squared Euclidean distance between the received point and the points along the transitions.

In our design we have chosen to use Read Only Memory's (ROM's) to store the precalculated BP (three bits are used to represent each parallel path) and BM (based on log-likelihood metrics). The encoder can produce one of eight (i.e., $2^{\tilde{k}+1}$) sets of parallel paths (each containing 8 paths). The BP and BM must be determined for each of these eight sets of parallel paths.

We have chosen four bits to represent the BM value. This gives a BM range from 0 (closest to the received 4D point) to 15 (furthest from the 4D point). Decoder simulations in [5] for another multi-D trellis code indicate that this amount of quantisation results in little performance degradation.

To minimise the number of address bits to the ROM, each received 2D signal point has been quantised to seven bits. After extensive simulations in [5] for a 6D-8PSK trellis code, it was found that pie-chart or angular quantisation results in the least performance degradation (0.2 to 0.3 dB for five bit quantisation). The simulations included the "dartboard" quantisation pattern proposed in [1].

Each ROM therefore has an address space of 14 bits (seven bits for each 2D symbol). The ROM's used for the BMC are 32K×8 27C256's. A total of 6 ROM's were used, two for determining the BP's and four for the eight BM's.

Alternative BMC schemes which exploit the finite length trellis structure of the parallel transitions were also considered. That is, a Viterbi like decoder can be used to decode the parallel transitions. However, their large complexity (in a discrete implementation) led us to choose the simpler ROM look-up method. For a VLSI implementation, though, the trellis decoding method would be preferable due to the flexibility that VLSI provides in designing circuits. Thus, the Viterbi decoder (with the BMC) could be implemented on a single chip.

3.2 State Metric Calculator

The SMC updates the State Metrics (SM) for each state of the code in each iteration of the Viterbi algorithm. A SM is an indication of how close the received sequence is to the closest path of all paths leading into a particular state. Since the code has two checked bits, there are four paths leading into each state (since we choose the closest path among the 8 parallel paths in the BMC). For each of the four paths, we must add the BM for that path to its corresponding SM (also known as the old SM) from the previous iteration. The new SM for the four paths leading into a state is the smallest of these summations. This path is selected and all other paths are eliminated. This is called the Add-Compare-Select (ACS) operation.

With four paths into each state a 4:1 ACS circuit is required. With 16 states in our code, the ACS operation needs to be performed 16 times (explaining the need for 16 clock cycles). The ACS circuit also produces two Path Decision (PD) bits which indicate which of the four paths was chosen. This information is passed to the SSM where it is stored.

Since the decoder operates serially, only one ACS circuit is required. The 16 SM's are stored in two 74AS870 dual 16×4 static Random Access Memory (RAM) chips. Eight bits are used to represent each SM. As shown in [5] for a 6D-8PSK trellis code, this is more than enough bits when two's complement arithmetic is used in the ACS circuit to prevent overflow [4]. Before the first new SM can be calculated, four old SM's are read out from the RAM's. This takes four clock cycles. It takes another two clock cycles to perform the ACS operation. To achieve a slightly higher speed, we could have done the ACS operation in one clock cycle. However, this would have required six comparator chips to find the minimum SM. An increase of one clock cycle and the use of three comparator chips was chosen to decrease the complexity of the design.

Another clock cycle is used to write to the other half of the dual 16×4 RAM's. Since all the read and ACS operations are pipelined, an additional 15 clock cycles are required to write the 15 remaining new SM's. In the next iteration of the algorithm we read from where the SM's were written in the previous iteration and write to where the old SM's had been stored. The process then repeats.

For the ACS circuit, the appropriate BM's must be added to the correct old SM's. Twelve quad 2:1 multiplexer chips and a copy of the convolutional encoder are needed to accomplish this task.

3.3 Survivor Sequence Memory

The SSM has two tasks. It must store the Path Decisions (PD's) generated by the SMC and "traceback" through the previously stored PD's to determine the final decoded bits for x^2 and x^1. This requires alternating write and read (for the traceback) operations on the memory. The traceback depth is the required number of PD sets (each set consists of 16 two bit PD's) that the SSM must trace back through.

The PD's must be stored in the remaining 16 clock cycles that are available. There are two ways this can be achieved. Storing two PD bits in each clock cycle or storing four PD bits in every other cycle, leaving the alternate cycle to perform part of the traceback. With the first method at least two separate memories are required since the traceback operation cannot be performed simultaneously with the storage of the new set of PD's (due to the design of memory chips). Since there is a finite amount of memory, the oldest PD set must be written over.

There is usually a point where one method is better than the other (in terms of the total memory size required) based on the number of clock cycles available and the traceback depth. A traceback depth of around 25 to 30 results in little performance degradation [5]. Comparing the implementation complexity of the two methods, the alternating read/write method proved superior.

With this design only eight clock cycles are available to perform a traceback. To maintain integer power of 2 address spaces for the memories (and thus efficiently use of practical memory designs), a traceback depth of seven is used for each SSM memory chip. To achieve the required traceback depth, four 64×4 memories are required. This gives a traceback depth of 28. The traceback is performed in a pipelined fashion, switching between memories when required and waiting for the next received set of data to continue with the traceback. Four separate memories are required since there are four tracebacks in operation at any one time.

Since there are no 64×4 RAM's commercially available, larger 256×4 93422A RAM's were used. This chip has separate input and output data buses which simplifies the SSM design. We use the state with the smallest SM to start the traceback. This is the best state the SSM could start with (since it corresponds to the path that is closest to the received signal) and helps give the decoder a slight performance improvement over choosing a random or a fixed state. The Minimum State Metric Selector (MSMS) provides the information needed to achieve this.

At the correct time and place in the circuit, the two decoded bits x^1 and x^2 are produced. The two bits are passed to the Branch Point Selector (BPS) where they are re-encoded to select one of the eight 3 bit branch points. The branch points are delayed by 34 4D symbol periods, 28 due to the traceback, 4 due to the pipeline delay in the traceback, and 2 due to the re-encoding of the decoded data.

The five decoded bits are then differentially decoded (optional) and then parallel to serial converted for the final decoder output. Precoding and postdecoding are optional as there are some communication systems that do not require phase synchronisation. For example, a burst modem can provide phase information in the preamble of a burst. A 74HC4046 PLL is used to generate the required five times clock for the serial data. This PLL is tuned to lock within 0 to 2 MHz, but as expected for PLL's the lower frequency limit will be somewhat greater than DC. The decoded data is also available in five bit bytes.

3.4 Signal Set Synchroniser

The SSS has the task of synchronising the decoder to the received sequence of 2D symbols. Since the signal set consists of two 2D signals, the decoder must synchronise to one of the two possible ways the received data can arrive.

The decoder is asynchronously locked to DATCLK, which is the received 2D symbol clock whose frequency has been divided by two. A delay of zero or one 2D symbol periods of DATCLK is used for timing synchronisation.

The SSS works by examining the rate of increase of the minimum SM from the MSMS. If the rate is high, this indicates that the decoder is out of synch and needs to be resynchronised. A variable threshold in the SSS is used for this purpose. If the threshold is exceeded, the SSS will toggle into the "arm symbol toggel" state.

If the threshold is again exceeded in the next V (V is a variable from 0 to 63) 4D symbol periods the decoder will toggle the 2D symbol delay (from zero to one or one to zero). The SSS then ignores the decoder for 128+V 6D symbol periods to allow the decoder to settle into its new signal set configuration.

If the threshold is not exceeded the SSS will "disarm" and return to its normal monitoring state.

4 OTHER DECODER FEATURES

The encoder and decoder are mounted within a 3U high 19 inch rack. On the front panel, two Light Emitting Diodes (LED's) are used to indicate the 2D symbol delay.

To test the decoder, the 2D symbol delay can be independently set to manual control. In this way, the SSS can be isolated from the rest of the circuitry so that any problems with the rest of the decoder can be fixed without the SSS interfering. It can also be used to test the SSS by manually introducing delays into the received signal. There are two switches used for this.

Two rotary type switches are used to select the format of the received data. One switch is used to select between 3 bit phase (corresponding to hard decision), 7 bit phase quantisation, 5 bit I and Q quantisation, or internal loopback mode. The other switch selects between signed magnitude, reverse binary, straight binary, or two's complement data formats for I and Q received data.

There are also switches for disabling the postdecoder from the decoder and the precoder from the encoder. The encoder has another switch to select between five bit parallel or bit serial data. The decoder also has a reset button to force all the SM's to zero. The encoder/decoder interface diagram is given in Figure 3.

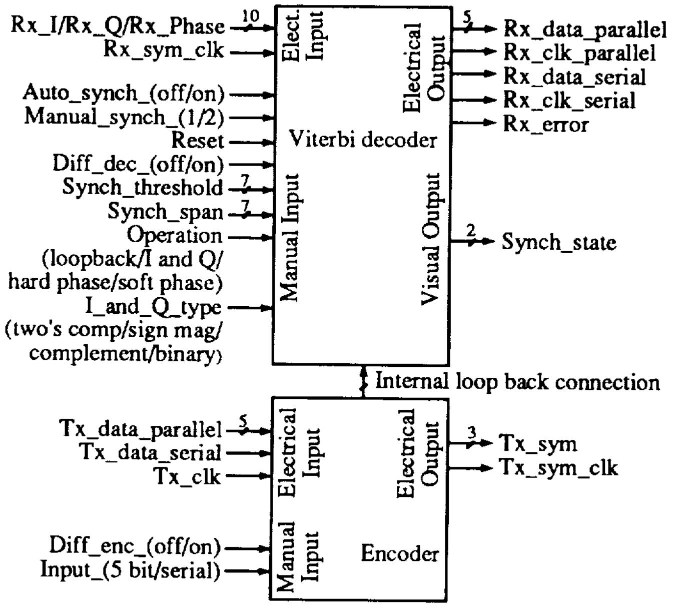


Figure 3: Viterbi decoder/encoder interface diagram for 16 state 2.5 bit/sym 4D-8PSK trellis code.

The 159 integrated circuits of the design are placed on two double height Speedwire Eurocards (233.4×220 mm). Speedwire allows quick and reliable connections (if it is done correctly) between the chips that can be easily changed. The speedwire boards also have good groundplanes, critical when operating at high clock speeds. The Viterbi decoder (which operates at 10 MHz) is placed on one board (taking 96 chips) while the encoder, SSS, and various interface chips are placed on the other board.

BNC connectors are used at the back of the rack for external data and clock connections. It is assumed that all received data changes on the rising edge of its clock. Similarly, the codec produces its signals in the same format. TTL 75 Ω interface signals are used for these external interfaces.

6 CONCLUSIONS

A serial implementation of a Viterbi decoder for the 16 state 2.5 bit/sym code with a 4D-8PSK signal set has been described. This decoder can provide high data rates (up to 2.1 Mbit/s) and is intended for future use on the Hubble Space Telescope. Due to its serial implementation the decoder design is quite complex, but could be implemented on a single VLSI integrated circuit.

The Branch Metric Calculator has been implemented through the use of large look-up table ROM's. A VLSI implementation may use a Viterbi type decoding algorithm to allow single chip implementation.

REFERENCES

[1] Pietrobon, S. S., R. H. Deng, A. Lafanechère, G. Ungerboeck, and D. J. Costello, Jr., "Trellis-coded multidimensional phase modulation," *IEEE Trans. Inform. Theory*, vol. 36, pp. 63-89, Jan. 1990.

[2] Perez, L., "On the performance of multi-dimensional phase modulated trellis codes," NASA Tech. Report #89-10-02, Oct. 1989.

[3] Porath, J. E., "Algorithms for converting convolutional codes from feedback to feedforward form and vice versa," *IEE Electron. Lett.*, vol.25, pp. 1008-1009, 20 Jul. 1989.

[4] Pietrobon, S. S., "Rotationally invariant convolutional codes for MPSK modulation and implementation of Viterbi decoders," M.Eng. Thesis, School of Electron. Eng., South Australian Inst. of Technol. (now University of South Australia), Adelaide, Australia, Jun. 1988.

[5] Gray, P. K., I. S. Morrison, and W. G. Cowley, "The development of a 45 Mbit/s modem/codec," *Proc. IREECON-'89*, pp. 942-945, Melbourne, Australia, Sep. 1989.